**­СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc184038900)

[ГЛАВА 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 6](#_Toc184038901)

[1.1 Создание базы данных и работа с ней в PHPMyAdmin 6](#_Toc184038902)

[1.2 Разработка API на языке программирования Python 8](#_Toc184038903)

[1.3 Интерфейс базы данных 11](#_Toc184038904)

[ГЛАВА 2 СПЕЦИАЛЬНАЯ ЧАСТЬ 15](#_Toc184038905)

[2.1 Постановка задачи 15](#_Toc184038906)

[2.2 Проектирование и реализация базы данных 16](#_Toc184038907)

[2.3 Разработка API 24](#_Toc184038909)

[2.4 Разработка веб-интерфейса для базы данных 31](#_Toc184038910)

[ГЛАВА 3 ЭКОНОМИЧЕСКАЯ ЧАСТЬ 36](#_Toc184038911)

[ГЛАВА 4 МЕРОПРИЯТИЯ ПО ТЕХНИКЕ БЕЗОПАСНОСТИ 48](#_Toc184038912)

[ЗАКЛЮЧЕНИЕ 50](#_Toc184038913)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 51](#_Toc184038914)

# ВВЕДЕНИЕ

Современное образование требует внедрения инновационных технологий для обеспечения качественного обучения студентов. Создание специализированного веб-ресурса для обучения английскому языку повысит конкурентоспособность выпускников, улучшит их навыки общения в международной среде, предоставит помощь учителям в индивидуальной работе с учащимися, обеспечит гибкость и доступность в обучении, а также будет соответствовать современным технологическим тенденциям, способствуя мотивации и эффективности процесса изучения языка.

Целью данного проекта является создание нескольких модулей для веб-ресурса, предназначенного для обучения английскому языку студентов специальности "Компьютерные системы и комплексы". В рамках работы будет разработана база данных с использованием утилиты PHPMyAdmin, пользовательский интерфейс в виде веб-страницы и прикладной программный интерфейс (API), обеспечивающий взаимодействие между базой данных и веб-страницей.

Основой структуры базы данных станет принцип модульности, что позволит преподавателям повторно использовать ранее созданные модули и темы, тем самым ускоряя процесс создания учебного материала.

# ГЛАВА 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1.1 Создание базы данных и работа с ней в PHPMyAdmin

PHPMyAdmin - это популярный и широко применимый инструмент в управлении базами данных MySQL. Он предоставляет удобный веб-интерфейс, что делает выполнение различных операций, включая создание, модификацию и удаление таблиц и записей, выполнение SQL-запросов и управление пользователями, более доступным и простым. Благодаря возможности работы в графическом интерфейсе, PHPMyAdmin облегчает выполнение даже самых сложных задач, повышая общую эффективность работы администратора базы данных. Кроме того, PHPMyAdmin обладает обширным функционалом, включающим такие возможности, как экспорт и импорт данных, создание резервных копий и их восстановление, а также мониторинг состояния сервера. Важным преимуществом является высокий уровень безопасности, обеспечиваемый поддержкой различных методов аутентификации и возможностью настройки прав доступа для пользователей.

Возможности, предоставляемые PHPMyAdmin, могут помочь на многих этапах создания и работы с базой данных:

1. Проектирование базы данных. На этапе проектирования определяются основные сущности, их атрибуты и связи между ними. PHPMyAdmin предоставляет возможности для создания и визуализации схем базы данных, что упрощает процесс проектирования. С помощью этого инструмента можно легко определять первичные и внешние ключи, а также задавать необходимые ограничения и индексы, что способствует поддержанию целостности данных и оптимизации запросов;
2. Создание базы данных и таблиц. После завершения проектирования структура базы данных воплощается в виде конкретных таблиц и их связей. PHPMyAdmin предоставляет удобные интерфейсы для создания таблиц, где можно определить типы данных для каждого столбца и настроить ключевые поля. Возможность использовать как графический интерфейс, так и SQL-запросы для создания таблиц обеспечивает гибкость и точный контроль над процессом;
3. Наполнение базы данных данными. Этот этап включает загрузку начальных данных в базу. PHPMyAdmin поддерживает импорт данных из различных форматов, таких как CSV и SQL-файлы, что облегчает перенос больших объемов информации в базу данных. Инструменты для массового ввода данных и удобные формы для ручного добавления записей позволяют быстро и эффективно наполнять таблицы необходимой информацией;
4. Оптимизация и настройка базы данных. Для обеспечения высокой производительности базы данных необходимо провести оптимизацию. PHPMyAdmin предоставляет средства для анализа производительности запросов и настройки индексов, что помогает ускорить выполнение часто используемых запросов. Поддержка хранимых процедур и триггеров позволяет автоматизировать часто выполняемые операции и повысить общую эффективность работы базы данных;
5. Управление и администрирование базы данных. Этот этап включает в себя регулярное администрирование, создание резервных копий и мониторинг состояния сервера. PHPMyAdmin предлагает функции для создания и восстановления резервных копий, что обеспечивает надежное хранение данных и возможность их восстановления в случае сбоя. Инструменты для мониторинга позволяют отслеживать производительность сервера и выявлять потенциальные проблемы. Также PHPMyAdmin предоставляет средства для управления пользователями и их правами доступа, что помогает защитить данные от несанкционированного доступа.

Таким образом, использование PHPMyAdmin на всех этапах работы с базой данных значительно упрощает процесс ее создания, управления и администрирования. Этот инструмент обеспечивает удобный интерфейс и мощные функции, которые помогают эффективно справляться с задачами любой сложности, от проектирования структуры данных до оптимизации производительности и обеспечения безопасности.

## 1.2 Разработка API на языке программирования Python

Прикладной программный интерфейс (API) представляет собой набор инструментов и протоколов, используемых для создания программного обеспечения и обеспечения взаимодействия различных приложений. API упрощает интеграцию между различными компонентами системы, обеспечивает стандартизированный обмен данными и улучшает общую производительность разработки, предоставляя доступ к данным и функциональным возможностям программного обеспечения без необходимости глубокого понимания их внутренней реализации.

В рамках данной работы прикладной программный интерфейс будет служить для связи базы данных и интерфейса в виде веб-сайта.

Python и PHP являются популярными языками программирования, используемыми для разработки API. Главным решением при разработке прикладного программного интерфейса был выбор между этими двумя языками.

Оба языка, как PHP, так и Python, являются интерпретируемыми. Это означает, что их код исполняется построчно интерпретатором, а не компилируется в машинный код заранее. Интерпретируемые языки обычно имеют более медленное выполнение программ по сравнению с компилируемыми, что может негативно сказываться на производительности. Однако их преимущество заключается в гибкости разработки, быстроте и удобстве отладки.

Популярность PHP объясняется его широким распространением и доступностью. PHP интегрируется с большинством веб-серверов и систем управления базами данных, что делает его удобным для веб-разработчиков. Он предлагает простоту использования и низкий порог входа, что позволяет начинающим разработчикам быстро начать работу. Благодаря своей гибкости и мощным функциям PHP позволяет создавать динамические веб-страницы и сложные веб-приложения. PHP имеет богатую экосистему с множеством доступных фреймворков и библиотек, таких как Laravel и Symfony, которые упрощают процесс разработки и позволяют создавать масштабируемые и поддерживаемые приложения. PHP также имеет активное сообщество, которое предоставляет множество ресурсов и инструментов для решения различных задач разработки.

Однако PHP имеет и недостатки. Одним из главных является менее строгая типизация по сравнению с другими языками, что может привести к ошибкам и проблемам при масштабировании приложений. Кроме того, код на PHP иногда может быть менее читаемым и сложным в поддержке, особенно в крупных проектах.

Python также является языком программирования, который широко используется для создания API. Одним из ключевых преимуществ Python является его простота и читаемость кода, что облегчает процесс разработки и последующего сопровождения программного обеспечения. В отличие от PHP, Python имеет строгую типизацию, что снижает вероятность ошибок и улучшает читаемость кода. Также этот язык поддерживает объектно-ориентированный и функциональный стили программирования, что дает разработчикам гибкость в выборе подхода к решению задач. Богатая стандартная библиотека Python и множество доступных сторонних библиотек делают его мощным инструментом для решения различных задач.

В данном проекте выбор был сделан в пользу Python, так как он обладает строгой типизацией, что снижает вероятность ошибок и улучшает читаемость и поддержку кода. Также Python обеспечивает более высокую производительность при разработке за счет удобства и скорости написания кода, что критично при создании и поддержке API.

В процессе разработки использовались инструменты MySQL Connector, Flask и Postman. MySQL Connector был выбран для взаимодействия с базой данных MySQL. Этот инструмент обеспечивает надежное и эффективное соединение с базой данных и обладает хорошей документацией, что делает его идеальным выбором для работы с MySQL. Альтернативами могли быть такие библиотеки, как SQLAlchemy и PyMySQL. SQLAlchemy предлагает более высокоуровневый доступ к базе данных с поддержкой ORM, но его использование было бы избыточным для данного проекта. PyMySQL также предоставляет соединение с MySQL, но MySQL Connector предпочтительнее из-за его производительности и стабильности.

Flask будет использован для обеспечения взаимодействия между Python кодом и веб-страницей, он был выбран в качестве фреймворка благодаря своей легковесности, гибкости и простоте использования. Flask предоставляет минималистичный подход к разработке, позволяя разработчикам добавлять только необходимые компоненты, что повышает контроль над архитектурой приложения. Альтернативами могли быть Django и FastAPI. Django — мощный фреймворк с большим количеством встроенных функций и админ-панелью, но его использование было бы излишне сложным для относительно небольшого API. FastAPI предлагает современные возможности асинхронного программирования и высокую производительность, однако Flask был выбран за его простоту и широкую документацию, что упрощает процесс разработки и обучения.

Postman использовался для тестирования и отладки API благодаря его интуитивно понятному интерфейсу и широким возможностям для автоматизации тестирования. Этот инструмент позволяет легко создавать, управлять и выполнять тесты API, что ускоряет процесс разработки и улучшает качество кода. Альтернативой мог быть такой инструмент, как Insomnia. Insomnia предлагает аналогичные функции для тестирования API, но Postman был выбран за его широкую распространенность и поддержку сообщества, что делает его более удобным и универсальным решением.

Таким образом, выбор MySQL Connector, Flask и Postman был обусловлен их надежностью, эффективностью и соответствием требованиям проекта. Эти инструменты позволяют создавать, тестировать и поддерживать API на высоком уровне, обеспечивая гибкость и удобство в процессе разработки.

В заключение, разработка API на языке программирования Python предоставляет множество преимуществ, включая удобство разработки, гибкость и поддержку сообщества. Использование таких инструментов, как MySQL Connector, Flask и Postman, позволяет создавать эффективные и надежные API, которые легко тестировать и поддерживать.

## 1.3 Интерфейс базы данных

При создании базы данных крайне важно обеспечить её удобный и функциональный интерфейс перед передачей заказчику. Интерфейс базы данных служит не только средством для получения пользователями данных, но и инструментом для выполнения других операций, таких как ввод, изменение и удаление. Отсутствие интерфейса может значительно усложнить работу с базой данных, особенно для пользователей, не обладающих глубокими знаниями в области программирования и администрирования баз данных.

С коммерческой точки зрения, интерфейс позволяет заказчику оценить функциональность и возможности базы данных. Наличие удобного и интуитивно понятного интерфейса демонстрирует заказчику, как данные могут быть эффективно использованы для поддержки бизнес-процессов. Это помогает визуализировать потенциальные выгоды и применение базы данных в реальной работе. Также интерфейс базы данных способствует более быстрому и легкому внедрению системы в рабочие процессы заказчика. Пользователи могут легко освоить новую систему, поскольку им не нужно разбираться в сложных технических аспектах работы с базой данных. Это сокращает время на обучение персонала и позволяет быстрее начать использовать систему в повседневной деятельности, что является значительным преимуществом, сильно повышающим ценность работы в глазах заказчика.

Таким образом, создание интерфейса является неотъемлемой частью процесса разработки базы данных, обеспечивающей её доступность и удобство использования.

Существует несколько основных вариантов создания интерфейса для базы данных, каждый из которых имеет свои преимущества и недостатки.

Первым вариантом является использование настольных приложений. Эти приложения предоставляют мощные возможности для управления данными и могут быть настроены под конкретные нужды заказчика. Настольные приложения обычно обладают высокой производительностью и не зависят от качества интернет-соединения. Однако их недостатком является сложность в установке и обновлении, особенно в средах с большим количеством пользователей. Кроме того, настольные приложения часто ограничены операционной системой, под которую они разработаны.

Другим вариантом является использование мобильных приложений. Такие приложения удобны для работы на ходу и обеспечивают доступ к базе данных из любого места с мобильным интернетом. Преимущества мобильных приложений включают удобство использования и возможность интеграции встроенных возможностей мобильных устройств, таких как камеры, геолокация и сенсоры, что может существенно расширить функциональные возможности базы данных.

Однако разработка мобильных приложений требует значительных ресурсов и времени, а также обязательного учета различных мобильных платформ, что может увеличивать сложность и стоимость проекта. Также нельзя забывать о том, что мобильные приложения могут быть полезны только как дополнительный вариант они не способны полностью заменить интерфейс для компьютера так ка ограниченный размер экрана и меньшая вычислительная мощность мобильных устройств могут затруднить выполнение сложных операций с базой данных.

Таким образом, мобильные приложения лучше рассматривать в качестве вспомогательного инструмента для пользователей, которые нуждаются в доступе к базе данных в условиях мобильности. Основное взаимодействие с базой данных, особенно в профессиональной и бизнес-среде, все же будет более эффективным через интерфейсы, разработанные для компьютеров.

Третьим вариантом является создание веб-интерфейса. Веб-интерфейс обеспечивает доступ к базе данных через веб-браузер, что делает его кроссплатформенным и не требует установки дополнительных программ. Преимущества веб-интерфейса включают легкость обновления и обслуживания, доступность из любой точки мира с интернет-соединением и возможность масштабирования под различные устройства. Основным недостатком веб-интерфейса может быть зависимость от качества интернет-соединения и потенциальные проблемы с безопасностью, если не обеспечены соответствующие меры защиты.

При анализе всех вариантов было принято решение в пользу создания веб-интерфейса для базы данных. Этот выбор обусловлен его кроссплатформенностью, удобством обновления и обслуживания, а также доступностью из любой точки с интернет-соединением. Веб-интерфейс позволяет обеспечить гибкость и масштабируемость, необходимые для современных информационных систем, и упрощает работу пользователей с базой данных, независимо от их технических навыков и используемых устройств.

При создании веб-интерфейса будет использован Bootstrap. Этот фреймворк позволит обеспечить адаптивный и привлекательный дизайн, улучшив визуальное восприятие и удобство использования интерфейса. Bootstrap предоставляет готовые стили и компоненты, что значительно ускоряет процесс разработки и обеспечивает единообразие внешнего вида элементов интерфейса.

В заключение, создание интерфейса для базы данных является критически важным этапом, обеспечивающим её удобство и доступность для конечных пользователей. Среди рассмотренных вариантов создания интерфейса, веб-интерфейс был выбран как наиболее подходящий.

Таким образом, в 1 главе были рассмотрены основные аспекты создания и управления базами данных с использованием PHPMyAdmin, причины выбора языка программирования Python для разработки API, связанные с этим особенности, а также назначение проектирования интерфейса базы данных и причины выбора конкретного вида реализации.

Для проектирования базы данных был выбран PHPMyAdmin, который предоставляет удобный веб-интерфейс для управления базами данных MySQL, упрощая выполнение различных операций. Рассмотрены возможности этого инструмента на всех этапах жизненного цикла базы данных, от проектирования до администрирования, было доказано, что PHPMyAdmin повышает общую эффективность работы администратора.

Разработка API на языке Python была выбрана из-за строгой типизации, простоты и читаемости кода, а также мощной стандартной библиотеки. В работе описаны инструменты MySQL Connector, Flask и Postman, которые были выбраны за надёжность, эффективность и соответствие требованиям проекта.

Создание интерфейса базы данных является важным этапом, обеспечивающим её доступность и удобство использования для конечных пользователей. Рассмотрены варианты интерфейсов, и обосновано, что веб-интерфейс является наиболее подходящим благодаря своей кроссплатформенности, лёгкости обновления и доступности из любой точки с интернет-соединением.

Таким образом, теоретическая часть дипломной работы охватывает ключевые аспекты работы с базами данных и API, обосновывая выбор используемых технологий и инструментов для создания эффективной, удобной и надёжной системы.

# 

# ГЛАВА 2 СПЕЦИАЛЬНАЯ ЧАСТЬ

## 2.1 Постановка задачи

Цель данного проекта заключается в разработке ряда модулей для веб-ресурса, предназначенного для обучения английскому языку студентов специальности “Компьютерные системы и комплексы”. В рамках проекта будет создана база данных с использованием утилиты PHPMyAdmin, разработан пользовательский интерфейс в виде веб-страницы, а также прикладной программный интерфейс (API), который обеспечит взаимодействие между базой данных и веб-страницей.

База данных будет предназначена для хранения информации о пользователях, учебных курсах и модулях, прогрессе каждого пользователя, а также словах, включая их синонимы и антонимы. Кроме того, в ней будет храниться информация об учебных упражнениях и их подтипах. В основе структуры базы данных будет лежать принцип модульности, который позволит преподавателю при составлении курсов повторно использовать ранее созданные модули и темы. Это значительно ускорит процесс создания учебного материала и обеспечит гибкость в организации учебного процесса.

Пользовательский интерфейс будет разработан с целью обеспечения комфортного и интуитивного взаимодействия пользователей с базой данных. Важной задачей интерфейса станет ограничение действий пользователя для предотвращения возможных ошибок и повреждений данных.

Прикладной программный интерфейс (API) послужит для упрощения и стандартизации взаимодействия между базой данных и пользовательским интерфейсом. Это позволит извлекать и вносить данные в базу данных без необходимости глубокого понимания внутренней реализации системы. API будет разработан с учётом заранее установленных правил и форматов обмена данными, что обеспечит надёжность и эффективность взаимодействия между компонентами системы.

Для тестирования и отладки разработанных компонентов будет подготовлен частичный набор данных для курса "Компьютерные системы и комплексы". Этот набор данных будет сформирован на основе методических документов, предоставленных преподавателем по иностранному языку, а также учебников, охватывающих соответствующие темы. Подготовка набора данных позволит провести всестороннее тестирование всех разработанных модулей и убедиться в их корректной работе.

## 2.2 Проектирование и реализация базы данных

Разработка базы данных началась с выполнения нормализации. Это позволит исключить избыточное дублирование данных, которое может стать причиной аномалий, возникающих при добавлении, редактировании и удалении строк таблицы, а также перерасхода памяти.

После приведения базы данных к третьей нормальной форме, таблицы имеют следующий вид:

* User (user\_ID, full\_name, email, hashed\_password, registration\_date);
* Course (course\_ID, title, description, page\_link);
* Course\_Participant (course\_participation\_ID, user\_id, course\_id, joining\_date, finishing\_date). Внешние ключи: user\_id, course\_id;
* Module (module\_ID, title, page\_link);
* Users\_Grade\_for\_Module (users\_grade\_for\_module\_ID, user\_id, module\_id, grade). Внешние ключи: user\_id, module\_id;
* Module\_in\_Course (module\_in\_course\_ID, course\_id, module\_id, order\_of\_the\_module, grade). Внешние ключи: module\_id, course\_id;
* Type\_of\_Exercise (type\_of\_exercise\_ID, name, description);
* Exercise (exersise\_ID, text, type\_of\_exercise\_id, answer). Внешний ключ: type\_of\_exercise\_id;
* Grammar\_Topic (grammar\_topic\_id, title, page\_link);
* Users\_Grade\_for\_Grammar\_Topic (users\_grade\_for\_grammar\_topic\_ID, user\_id, grammar\_topic\_id, grade). Внешние ключи: user\_id, grammar\_topic\_id;
* Grammar\_Topic\_of\_Exercise (grammar\_topic\_of\_exercise\_ID, exercise\_id, grammar\_topic\_id). Внешние ключи: exercise\_id, grammar\_topic\_id;
* Grammar\_Topic\_in\_Module (grammar\_topic\_in\_module\_ID, module\_id, grammar\_topic\_id, ). Внешние ключи: grammar\_topic\_id, module\_id;
* Word (word\_ID, word, Russian\_translation, English translation);
* Users\_Grade\_for\_Word (users\_grade\_for\_word\_ID, user\_id, word\_id, grade). Внешние ключи: user\_id, word\_id;
* Word\_in\_Exercise (word\_in\_exercise\_ID, exercise\_id, word\_id). Внешние ключи: exercise\_id, word\_id;
* Synonym (synonym\_ID, word\_id, word\_synonym). Внешний ключ: word\_id;
* Antonym (antonym\_ID, word\_id, word\_antonym). Внешний ключ: word\_id;
* Topic\_of\_Words (topic\_of\_words\_ID, name, description);
* Word\_in\_Topic (word\_in\_topic\_ID, word\_id, topic\_of\_words\_id). Внешние ключи: word\_id, topic\_of\_words\_id;
* Topic\_of\_words\_in\_Module (topic\_of\_words\_in\_module\_ID, module\_id, topic\_of\_words\_id). Внешние ключи: module\_id, topic\_of\_words\_id.

Следующим этапом является описание структуры основных таблиц.

Таблица “User” (пользователь) служит для хранения информации о пользователе. В ней предполагается хранение пароля в виде хэша для большей надёжности (таблица 1).

Таблица 1 - Структура таблицы “User” (пользователь)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Тип поля | Размер поля | Обязательное поле | Ключевое поле |
| user\_ID | Числовое | Длинное число | Да | Да |
| full\_name | Текстовое | 255 | Да | Нет |
| email | Текстовое | 255 | Нет | Нет |
| hashed\_password | Текстовое | 255 | Да | Нет |
| registration\_date | Временное | - | Нет | Нет |

Таблица “Course” (курс) содержит информацию об учебных курсах. Поле “page\_link” предполагает запись ссылки на страницу ознакомления с спецификой курса (таблица 2).

Таблица 2 - Структура таблицы “Course” (курс)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Тип поля | Размер поля | Обязательное поле | Ключевое поле |
| course\_ID | Числовое | Длинное число | Да | Да |
| title | Текстовое | 255 | Да | Нет |
| description | Текстовое | 255 | Нет | Нет |
| page\_link | Текстовое | 255 | Нет | Нет |

Таблица “Course\_Participation” (участие в курсе) является вспомогательной и служит для избежания связи многие-ко-многим между таблицами “Course” (курс) и “User” (пользователь), один пользователь может состоять на нескольких курсах и в курсе может быть много разных пользователей (таблица 3).

Последующие вспомогательные таблицы выполняют похожую роль и не будут детально описываться в данном документе.

Таблица 3 - Структура таблицы “Course\_Participation” (участие в курсе)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Тип поля | Размер поля | Обязательное поле | Ключевое поле |
| course\_participation\_ID | Числовое | Длинное число | Да | Да |
| user\_id | Числовое | Длинное число | Да | Нет |
| course\_id | Числовое | Длинное число | Да | Нет |
| joining\_date | Временное | - | Нет | Нет |
| finishing\_date | Временное | - | Нет | Нет |

Таблица “Module” (модуль) предназначена для хранения информации об учебных модулях. Модули - это этапы прохождения курса, состоящие из грамматических и словарных упражнений (таблица 4).

Таблица 4 - Структура таблицы “Module” (модуль)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Тип поля | Размер поля | Обязательное поле | Ключевое поле |
| module\_ID | Числовое | Длинное число | Да | Да |
| title | Текстовое | 255 | Да | Нет |
| page\_link | Текстовое | 255 | Нет | Нет |

Таблица “Type\_of\_Exercise” предназначена для хранения названий разных типов упражнений и их описания (таблица 5).

Таблица 5 - Структура таблицы “Type\_of\_Exercise” (тип упражнения)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Тип поля | Размер поля | Обязательное поле | Ключевое поле |
| type\_of\_exercise\_ID | Числовое | Длинное число | Да | Да |
| name | Текстовое | 255 | Да | Нет |
| description | Текстовое | 255 | Нет | Нет |

Таблица “Exercise” (упражнение) предназначена для хранения данных о упражнениях (таблица 6). В поле “text” будет храниться текст упражнения с использованием специальных знаков, которые будут позже обрабатываться отдельно (например, знаки, указывающие на место, куда нужно подставить пропущенное слово).

Таблица 6 - Структура таблицы “Exercise” (упражнение)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Тип поля | Размер поля | Обязательное поле | Ключевое поле |
| exercise\_ID | Числовое | Длинное число | Да | Да |
| text | Текстовое | - | Да | Нет |
| type\_of\_exercise\_id | Числовое | Длинное число | Да | Нет |
| answer | Текстовое | 255 | Да | Нет |

Таблица “Grammar\_Topic” (граммастическая тема) предназначена для хранения информации о грамматических темах. Деление упражнений на подобные категории помогут в ускорении создания новых курсов за счёт модульности, то есть при создании нового курса не будет необходимости создавать и выбирать упражнения снова, можно будет использовать модули и темы из прошлых курсов (таблица 7).

Таблица 7 - Структура таблицы “Grammar\_Topic” (грамматическая тема)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Тип поля | Размер поля | Обязательное поле | Ключевое поле |
| grammar\_topic\_id | Числовое | Длинное число | Да | Да |
| title | Текстовое | 255 | Да | Нет |
| page\_link | Текстовое | 255 | Нет | Нет |

Таблица “Word” (слово) будет содержать информацию о словах. В ней хранится не только русский, но и английский перевод, это позволит повысить качество и разнообразие обучения (таблица 8).

Таблица 8 - Структура таблицы “Word” (Слово)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Тип поля | Размер поля | Обязательное поле | Ключевое поле |
| word\_ID | Числовое | Длинное число | Да | Да |
| word | Текстовое | 255 | Да | Нет |
| Russian\_translation | Текстовое | 255 | Да | Нет |
| English\_translation | Текстовое | 255 | Да | Нет |

Таблицы “Synonym” (синоним) и  “Antonym” (антоним) также послужат для повышения качества и разносторонности обучения. Эти данные было необходимо выделить из таблицы “Word” так как каждое слово может иметь разное количество синонимов и антонимов (таблицы 9 и 10).

Таблица 9 - Структура таблицы “Synonym” (синоним)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Тип поля | Размер поля | Обязательное поле | Ключевое поле |
| synonym\_ID | Числовое | Длинное число | Да | Да |
| word\_id | Числовое | Длинное число | Да | Нет |
| word\_synonym | Текстовое | 255 | Да | Нет |

Таблица 10 - Структура таблицы “Antonym” (антоним)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Тип поля | Размер поля | Обязательное поле | Ключевое поле |
| antonym\_ID | Числовое | Длинное число | Да | Да |
| word\_id | Числовое | Длинное число | Да | Нет |
| word\_antonym | Текстовое | 255 | Да | Нет |

Таблица “Topic\_of\_Words” (словарная тема) служит для модульности учебного материала (таблица 11).

Таблица 11 - Структура таблицы “Topic\_of\_Words” (словарная тема)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Тип поля | Размер поля | Обязательное поле | Ключевое поле |
| topic\_of\_words\_ID | Числовое | Длинное число | Да | Да |
| name | Текстовое | 255 | Да | Нет |
| description | Текстовое | 255 | Нет | Нет |

В итоге проектирования базы данных получилась следующая схема данных (рисунок 1):

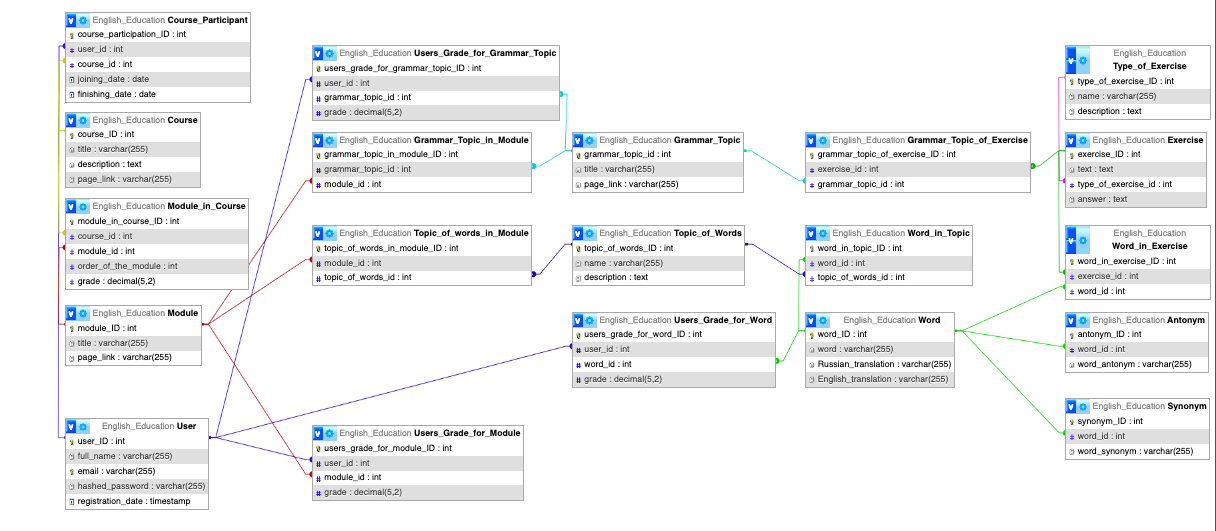


Рисунок 1 - Схема данных

## Для удобства взаимодействия с базой данных были заранее заготовлены комплексные SQL-запросы, которые помогут быстро получить необходимую информацию из объединения таблиц. Так, например, для анализа успеваемости пользователей был разработан запрос, позволяющий получить среднюю оценку пользователя по всем курсам. Другим примером является запрос для получения списка всех участников курса с подробной информацией о курсе и пользователе, что позволит администраторам легко отслеживать участие и прогресс пользователей в различных курсах. Кроме этих запросов были разработаны и многие другие.

В результате работы была разработана база данных, приведённая к третьей нормальной форме, что позволило исключить избыточное дублирование данных и предотвратить аномалии. Были созданы таблицы для хранения информации о пользователях, курсах, участниках курсов, модулях, типах упражнений, упражнениях, грамматических темах, словах, синонимах, антонимах и темах слов. Для удобства взаимодействия с базой данных были разработаны комплексные SQL-запросы, которые позволяют быстро получать необходимую информацию.

## 2.3 Разработка API

Основой для создания данного прикладного программного интерфейса (API) стал микрофреймворк Flask и библиотека MySQL Connector.

Задачи, за которые будет отвечать Flask:

* обработка HTTP-запросов. Flask принимает входящие HTTP-запросы от клиентов и маршрутизирует их к соответствующим обработчикам. В зависимости от типа запроса (GET, POST, PUT, DELETE) и URL, Flask вызывает соответствующую функцию для обработки запроса;
* форматирование ответов. Flask формирует ответы в формате JSON, которые возвращаются клиенту;
* обработка шаблона index.html, который является основной страницей сайта;
* обработка исключений и ошибок. Flask будет обрабатывать исключения и возвращать пользователю соответствующие сообщения об ошибках. Это повысит надёжность API и улучшает взаимодействие с пользователями;
* маршрутизация, связывания URL с конкретными функциями обработки запросов, что позволяет управлять различными аспектами взаимодействия с базой данных и предоставлять необходимую информацию клиентам.

Задачи, за которые будет отвечать MySQL Connector:

* управление подключением к базе данных. MySQL Connector устанавливает и поддерживает соединение с базой данных MySQL. Он обрабатывает все детали соединения, включая аутентификацию и выбор схемы базы данных;
* выполнение SQL-запросов к базе данных;
* преобразование полученных данных в удобный для использования формат;
* обработка ошибок, связанных с базой данных.

Разделение задач между Flask и MySQL Connector позволяет эффективно управлять различными аспектами работы API. Flask отвечает за взаимодействие с клиентами, обработку запросов и формирование ответов, тогда как MySQL Connector занимается взаимодействием с базой данных, выполнением запросов и управлением транзакциями. Такое разделение задач обеспечивает модульность и гибкость системы, позволяя легко масштабировать и поддерживать приложение.

Функция check\_connection проверяет текущее состояние соединения с базой данных. Если соединение отсутствует или прервано, функция пытается восстановить его, вызывая create\_connection. Также проверяется наличие курсора, который используется для выполнения запросов. При отсутствии курсора создается новый. В случае возникновения ошибок при проверке или восстановлении соединения выводится соответствующее сообщение.

Для подключения к базе данных используется функция create\_connection, которая создает соединение с MySQL и инициализирует курсор для выполнения запросов. Такая реализация позволяет централизованно управлять соединением и обеспечивает возможность его повторного установления в случае разрыва.

Данные для подключения к базе данных записаны в виде словаря:

config = {

'user': 'root',

'host': 'localhost',

'port': 3310,

'unix\_socket': '/Applications/XAMPP/xamppfiles/var/mysql/mysql.sock',

'database': 'English\_Education'

}

Функция insert\_data добавляет новую запись в указанную таблицу. Для наглядности общего принципа взаимодействия с MySQL Connector данная функция будет разобрана поэлементно. В первую очередь функция принимает название таблицы и её содержание:

def insert\_data(table, data):

global my\_cursor, connection

check\_connection()

Следующим шагом используется конструкция для проверки на ошибки при выполнении кода:

try:

...

except Error as e:

print(f"Ошибка при вставке данных: {e}")

raise

Внутри блока try сначала ключи словаря разбиваются на названия колонок, после создаётся SQL-запрос о вводе данных в таблицу:

try:

columns = ', '.join(data.keys())

placeholders = ', '.join(['%s'] \* len(data))

query = f"INSERT INTO {table} ({columns}) VALUES ({placeholders})"

Последним шагом запрос выполняется и изменения сохраняются в базе данных:

try:

...

my\_cursor.execute(query, tuple(data.values()))

connection.commit()

Функция fetch\_tables извлекает список таблиц из базы данных. Для этого она выполняет SQL-запрос SHOW TABLES, который возвращает все таблицы, содержащиеся в базе данных. Результат запроса обрабатывается и возвращается в виде списка имен таблиц.

Функция fetch\_raw\_data извлекает данные из указанной таблицы. При этом она также может сортировать данные по заданному столбцу в указанном порядке (по возрастанию или убыванию). Функция выполняет несколько SQL-запросов для получения информации о столбцах таблицы и самих данных. В случае отсутствия данных возвращается пустой список.

Особое внимание уделено функции replace\_foreign\_keys\_in\_data, которая заменяет значения внешних ключей на их соответствующие имена. Это значительно улучшает читаемость данных и упрощает их использование на стороне клиента.

Функция replace\_foreign\_keys\_in\_data проверяет каждый столбец на наличие внешнего ключа. Если столбец является внешним ключом, выполняется запрос к соответствующей таблице для получения имени записи по значению ключа. Если операция завершена успешно, возвращаются обновлённые данные.

Функция update\_data обновляет существующую запись в указанной таблице. Перед выполнением обновления проверяется наличие внешних ключей и при необходимости их значения заменяются на соответствующие идентификаторы. Формируется SQL-запрос для обновления данных по заданному условию, который выполняется с помощью курсора. После выполнения запроса изменения фиксируются в базе данных.

Функция delete\_data удаляет запись из указанной таблицы по значению заданного поля. Формируется SQL-запрос для удаления записи, который выполняется с помощью курсора. После выполнения запроса изменения фиксируются в базе данных.

Функция get\_primary\_key получает имя первичного ключа для указанной таблицы. Результат запроса возвращается в виде строки.

Функция get\_foreign\_key\_id получает значение внешнего ключа по имени записи в указанной таблице. Для этого сначала определяется имя первичного ключа таблицы, затем выполняется SQL-запрос для получения значения первичного ключа по имени записи. Результат запроса возвращается в виде значения идентификатора.

Все перечисленные функции разработаны с учетом универсальности, что позволяет использовать их для работы с любой таблицей в данной базе данных. Независимо от структуры и содержимого таблиц, эти функции обеспечивают надёжное и эффективное выполнение операций по извлечению, вставке, обновлению и удалению данных, а также обработке внешних ключей. Это делает их важным инструментом для создания гибкого и масштабируемого API.

API включает несколько маршрутов для выполнения различных операций с данными в базе данных. Каждый маршрут соответствует определенному методу HTTP-запроса (GET, POST, PUT, DELETE) и выполняет конкретные действия.

Главная страница приложения доступна по маршруту '/' и возвращает HTML-шаблон.

Маршрут '/tables' позволяет получить список всех таблиц в базе данных. Запрос к этому маршруту возвращает JSON-объект с именами таблиц.

Маршрут '/table/<table>' используется для получения данных из указанной таблицы. Запрос может включать параметры для сортировки данных и замены значений внешних ключей на соответствующие имена. Если данные успешно извлекаются, возвращается JSON-объект с записями, именами столбцов и информацией о столбцах, допускающих значения NULL.

Выбор использования сортировки при извлечении данных объясняется необходимостью предоставить пользователям гибкий инструмент для получения информации, учитывающий различные сценарии использования данных.

Для лучшего понимания принципов работы с микрофреймворком Flask этот маршрут будет разобран. В первую очередь название таблицы получается из url (значение должно быть записано на месте “<table>”) и передаётся в функцию add\_record:

@app.route('/table/<table>', methods=['POST'])

def add\_record(table):

Данные получаются в формате json:

data = request.json

В последней части функции используется конструкция для отлавливания ошибок при выполнении кода.

После использования функции insert\_data, описанной ранее, программа выдаёт обратную связь для пользователя:

try:

insert\_data(table, data)

return jsonify({"message": "Данные успешно записаны в базу данных"})

except Exception as e:

return jsonify({"error": str(e)}), 400

Маршрут '/table/<table>' с методом POST позволяет добавить новую запись в указанную таблицу. Данные для вставки передаются в формате JSON.

Маршрут '/table/<table>/<primary\_key\_value>' с методом DELETE используется для удаления записи из таблицы по значению первичного ключа.

Маршрут '/table/<table>/<primary\_key\_value>' с методом PUT позволяет обновить существующую запись в таблице по значению первичного ключа. Данные для обновления передаются в формате JSON.

Маршрут '/get\_foreign\_id/<table>/<name>' используется для получения значения внешнего ключа по имени записи. Если значение ключа найдено, возвращается JSON-объект с идентификатором.

Маршруты API включили в себя обработку различных типов HTTP-запросов (GET, POST, PUT, DELETE), что обеспечивает полный набор CRUD операций для управления данными в базе.

Использование методов HTTP-запросов позволило чётко разграничить различные операции над данными, следуя RESTful архитектуре, что улучшает структурированность и удобство использования API.

В процессе разработки API применялись проверенные подходы для обеспечения надёжности и удобства работы с базой данных. Flask был выбран за его простоту и гибкость, а библиотека MySQL Сonnector — за эффективные средства взаимодействия с MySQL. Особое внимание уделялось реализации ключевых функций, таких как замена внешних ключей и сортировка данных, чтобы повысить удобство и эффективность работы с данными для конечных пользователей.

## 2.4 Разработка веб-интерфейса для базы данных

Для разработки веб-интерфейса, позволяющего взаимодействовать с базой данных, использован HTML в сочетании с библиотеками Bootstrap, jQuery и Font Awesome. Данный интерфейс обеспечивает отображение таблиц, управление записями и дополнительные настройки для удобства работы с данными.

HTML был использован для создания структуры веб-страницы, включающей элементы управления, такие как выпадающие списки, таблицы, поля ввода и кнопки. Он задает основу, на которой строится весь интерфейс.

Bootstrap применялся для стилизации и создания адаптивного дизайна. С его помощью был обеспечен общий стиль и привлекательный внешний вид элементов интерфейса, таких как кнопки, таблицы и формы. Благодаря Bootstrap интерфейс корректно отображается на устройствах с различными размерами экрана.

jQuery использовался для управления событиями и выполнения AJAX-запросов. Он обеспечивает динамическое обновление содержимого страницы без необходимости перезагрузки. С помощью jQuery были реализованы функции выбора таблицы, добавления, редактирования и удаления записей, а также сортировки и фильтрации данных.

Font Awesome был применен для добавления иконок к элементам управления. Иконки улучшили визуальное восприятие интерфейса, сделав его более интуитивно понятным. Они использовались в кнопках добавления, редактирования и удаления записей, а также в элементах поиска и сортировки.

Использование этих инструментов в совокупности позволило создать функциональный и удобный веб-интерфейс для взаимодействия с базой данных, обеспечив высокий уровень удобства и эффективности работы с данными.

Боковая панель содержит элементы управления для настройки отображения данных и взаимодействия с таблицами базы данных. Включает флажки для включения и отключения функций добавления записей, показа таблиц с мета данными и замены внешних ключей на соответствующие имена записей, также есть возможность информации по заранее заготовленным сложным запросам (рисунок 2).

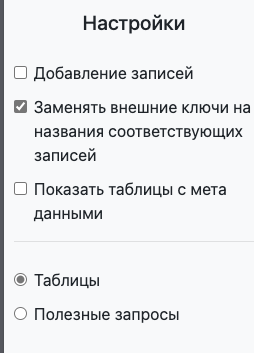


Рисунок 2 - Боковая панель

Таблицы с мета данными - это те таблицы, которые служат связью между таблицами, где необходима связь “многие ко многим”. В режиме по умолчанию таблицы с мета данными не показываются так как большинство пользователей не будет работать с ними на прямую.

Используя выпадающий список, пользователь может выбрать конкретную таблицу из базы данных для отображения и взаимодействия (рисунок 3).

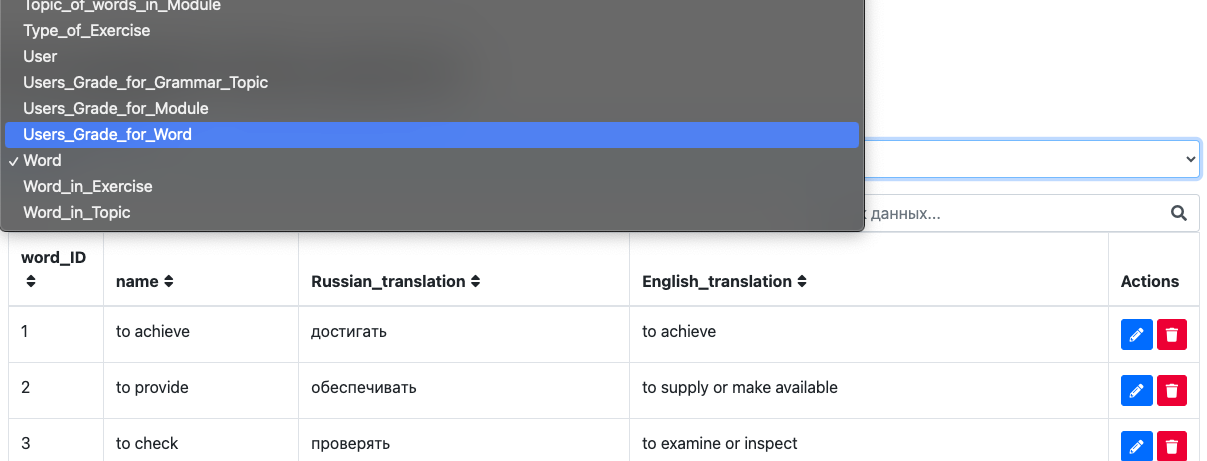


Рисунок 3 - Выбор таблиц для взаимодействия

Секция, предназначенная для добавления новых записей в выбранную таблицу, включает в себя таблицу с полями для ввода данных и кнопку для подтверждения добавления записи (Рисунок 4). Секция может быть скрыта или отображена в зависимости от настроек в боковой панели. После отправления данных для пользователя выводится сообщение о соответствующем результате.

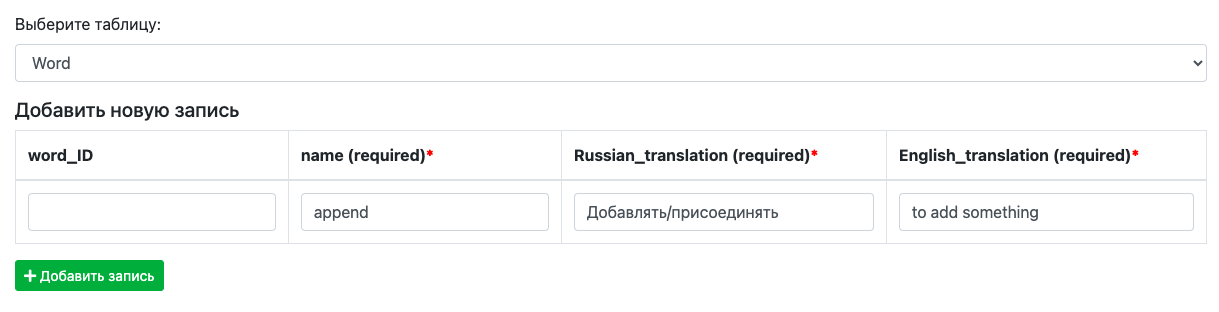


Рисунок 4 - Добавление новых записей

Рядом с выведенной таблицей имеется поле для ввода поисковых запросов, позволяющее фильтровать отображаемые данные в таблице. По мере ввода текста таблица автоматически обновляется, отображая только те строки, которые соответствуют введённому запросу (рисунок 5).

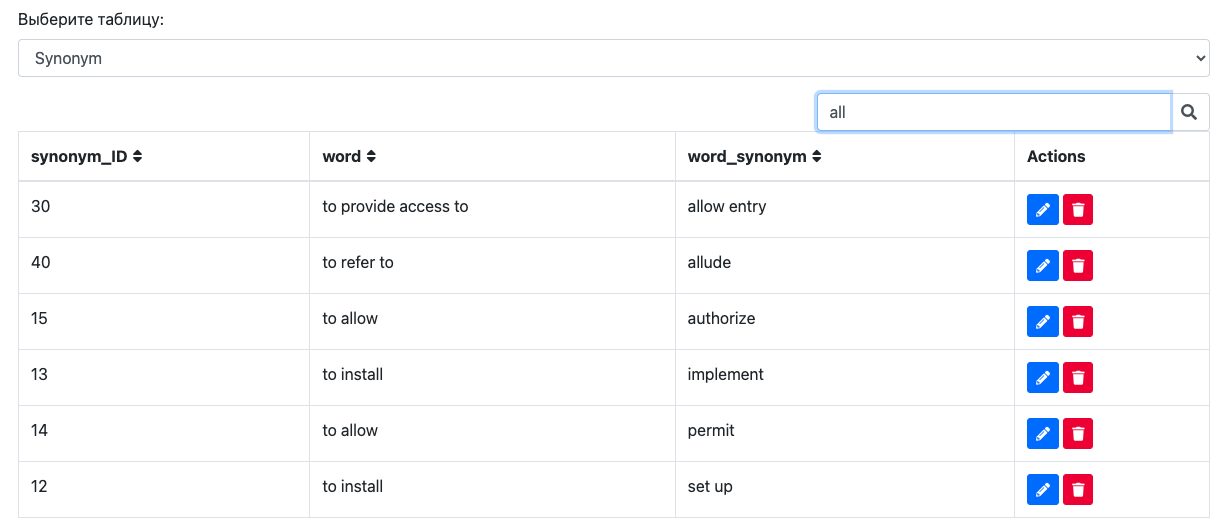


Рисунок 5 - Поисковая строка

В реализованном веб-интерфейсе есть функция, позволяющая сортировать данные в таблице по выбранному столбцу (рисунок 6). Пользователь может кликнуть на заголовок столбца, чтобы изменить порядок сортировки (по возрастанию или убыванию). Сортировка применяется к текущим отображаемым данным.

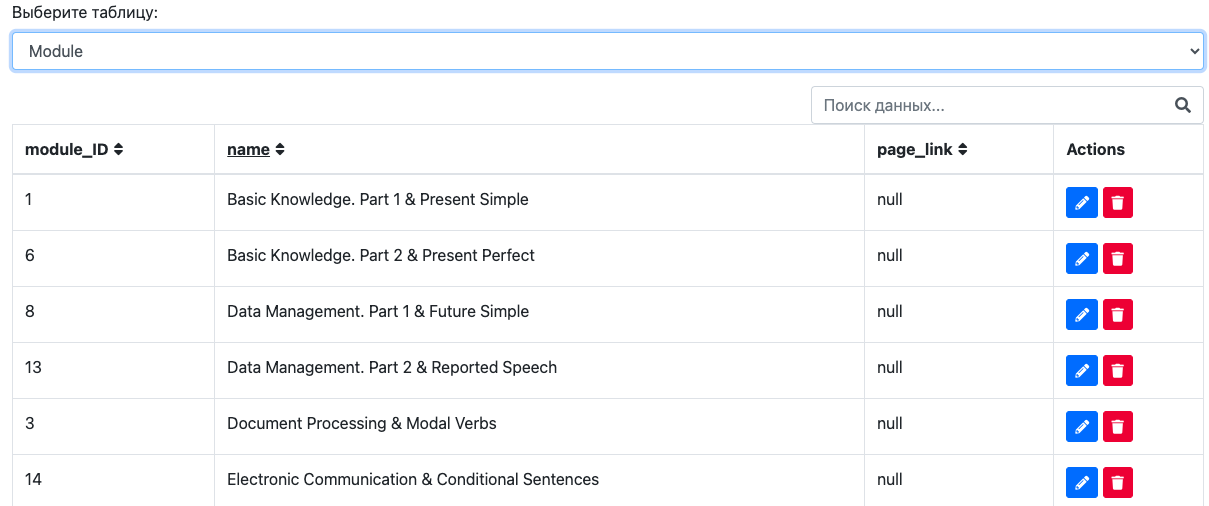


Рисунок 6 - Сортировка по столбцу

Также имеются функции, предоставляющие возможность редактирования и удаления существующих записей в таблице (рисунок 7). При нажатии на кнопку редактирования пользователь может изменить значения полей прямо в таблице, после чего обновлённые данные отправляются на сервер и, при условии допустимости значений, заносятся в базу данных. При нажатии на кнопку удаления записи и подтверждения нажатия, запись удаляется.

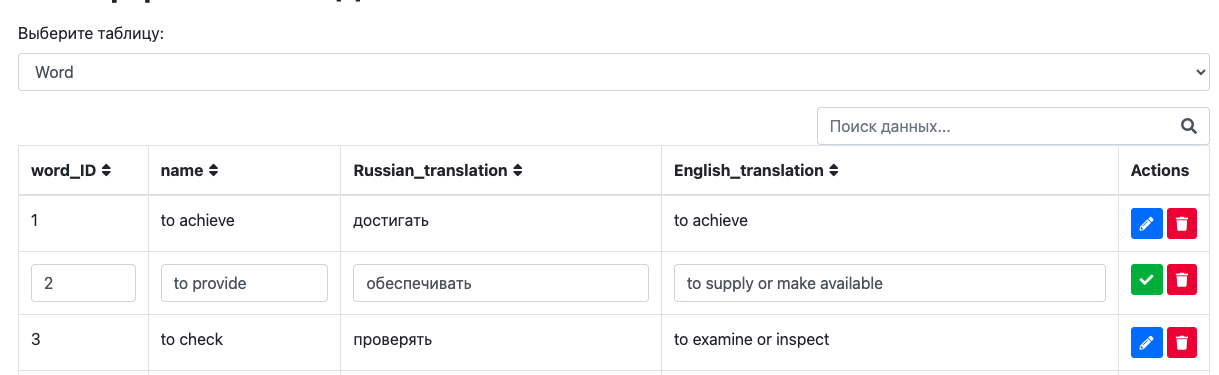


Рисунок 7 - Функции редактирования и удаления записей

Разработанный веб-интерфейс обеспечивает эффективное взаимодействие с базой данных. Он включает в себя удобные функции для управления записями, отображения таблиц и выполнения поисковых запросов. Система динамического обновления контента, адаптивный дизайн и интуитивно понятные элементы управления значительно улучшают удобство и эффективность работы с данными.

# ГЛАВА 3 ЭКОНОМИЧЕСКАЯ ЧАСТЬ

В качестве объекта дипломной работы для веб-ресурса по обучению английскому языку студентов специальности “Компьютерные системы и комплексы” создавалась база данных, её пользовательский интерфейс в виде веб-страницы, а также прикладной программный интерфейс для их взаимодействия.

Заказчиком является ГБПОУ “Донецкий техникум промышленной автоматики имени А.В.Захарченко”, образовательное учреждение среднего профессионального образования, расположенное в городе Донецк, специализирующееся на подготовке квалифицированных специалистов в различных областях. Целью готового сайта является создание эффективного инструмента, который повысит конкурентоспособность выпускников за счёт улучшения их навыков общения и понимания в международной среде, обеспечит дополнительную помощь учителям в индивидуальной работе с учащимися, предоставит студентам гибкость и доступность в обучении, а также будет соответствовать современным технологическим тенденциям, способствуя мотивации и эффективности процесса изучения языка.

В качестве конкурентов можно рассматривать многие онлайн-платформы для изучения английского языка, а также специализированные площадки для обучения программистов. Главным преимуществом, которым обладает сайт, для которого происходит разработка базы данных, пользовательского и прикладного программного интерфейса, является специализированность под специальности, по которым происходит обучение в техникуме. Этот сайт будет содержать учебные материалы и ресурсы, разработанные с учётом специфики каждой из этих областей. Студенты смогут изучать наиболее важные области английского языка для своей профессиональной деятельности, что сделает процесс обучения более эффективным и актуальным для будущей карьеры.

Для технико-экономического обоснования будет выполнены следующие расчёты: определение трудоёмкости работ, расчёт себестоимости разработки, определение цены и маркетинговый анализ.

Расчёт рабочего и машинного времени для каждого этапа работы представлен в таблице 12.

Таблица 12 - Расчёт трудоёмкости

|  |  |  |
| --- | --- | --- |
| Виды работ | Затраты времени, час | Машинное время работы над программным продуктом, час |
| Разработка технического задания | 4 | 3 |
| Сбор информации из методических документов и книг | 20 | 16 |
| Обработка и анализ собранного материала для последующего использования в ходе тестирования | 15 | 12 |
| Выбор программных средств разработки | 6 | 4 |
| Разработка первичной структуры базы данных | 12 | 7 |
| Нормализация базы данных | 8 | 6 |
| Создание базы данных в PHPMyAdmin | 12 | 8 |
| Разработка прикладного программного интерфейса | 20 | 15 |
| Разработка пользовательского интерфейса | 30 | 25 |
| Продолжение таблицы 12 |  | |
| Виды работ | Затраты времени, час | Машинное время работы над программным продуктом, час |
| Отладка | 10 | 8 |
| Подготовка технической документации и сдача продукта | 8 | 6 |
| Итого: | 145 | 110 |

Таким образом, на разработку данной программы было затрачено 145 часов человеческого времени и 110 часов машинного. Эти данные будут использованы для определения затрат на заработную плату, амортизацию оборудования и электроэнергию.

Расчёт себестоимости программы включает в себя следующие статьи затрат: сумма амортизационных отчислений, затраты на заработную плату, электроэнергию, общехозяйственные и прочие расходы.

Согласно статье 256 Налогового Кодекса Российской Федерации устройство относится ко второй группе основных средств, если его стоимость превышает 40000 рос. руб. Срок полезного использования для данной группы основных средств составляет от 2 до 3 лет.

Ликвидационная стоимость для основных средств второй группы составляет 5 процентов от начальной цены.

В ходе выполнения работы был использован компьютер, купленный за сумму, равную 51990 рос. руб. Данное компьютерное средство относится ко второй группе основных средств и обладает сроком полезного использования равным 3 годам.

Также был использован монитор стоимостью 43000 рос. руб. Срок полезного использования данного средства также составляет 3 года.

Годовая сумма амортизации определяется прямолинейным методом по формуле:

,

| где |  | - | первичная цена; |
| --- | --- | --- | --- |
|  |  | - | ликвидационная стоимость; |
|  |  | - | срок полезного использования в годах. |

Амортизационные отчисления за 1 месяц () определяются через деление на 12 месяцев.

Сумма амортизации по объекту основных средств, которые использовались в процессе создания дипломного проекта (), вычисляется по формуле:

,

| где |  | - | количество часов, затраченных на разработку с использованием компьютера; |
| --- | --- | --- | --- |
|  | МФ | - - | месячный фонд рабочего времени. |

Месячный фонд рабочего времени равен 175 часам. Расчёт амортизации представлен в таблице 13.

Таблица 13 - Расчёт амортизации основных средств

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Наименование основных средств | Количество, ед | Первоначальная стоимость, рос. руб. | Ликвидационная стоимость, рос. руб. | Годовые амортизационные отчисления на единицу, рос. руб. | Амортизационные отчисления на единицу за месяц, рос. руб. | Месячный фонд, час | Время использования, час | Суммарные амортизационные отчисления на период проведения, рос.руб |
| Компьютер | 1 | 51 990 | 2 599,5 | 16 463,5 | 1 371,96 | 175 | 110 | 862,37 |
| Монитор | 1 | 43 000 | 2 150 | 13 616,67 | 1 134,72 | 175 | 110 | 713,25 |
| Всего | | | | | | | | 1 575,62 |

В разработке программного изделия в условиях дипломного проектирования принимают участие разработчики трёх категорий:

* руководитель дипломного проекта;
* консультант дипломного проекта;
* дипломник.

Руководитель дипломного проекта является специалистом высшей категории с почасовой оплатой равной 396,46 рос. руб. Общее затраченное на консультации время составляет 16 часов.

Ставка взноса в социальный фонд Российской Федерации согласно пункту 3 статьи 425 Налогового Кодекса Российской Федерации составляет 30%.

Согласно списку, утверждённому приказом Минтруда от 30.12.2016 №851н, образовательная деятельность относится к I классу профессионального риска, для которого ставка взноса по травматизму составляет 0,2%.

Затраты на зарплату руководителя дипломного проекта () определяются по следующей формуле:

|  |  |  |  |
| --- | --- | --- | --- |
| где |  | - | размер почасовой заработной платы руководителя дипломного проекта; |
|  |  | - | количество часов, которые затрачивает руководитель на консультирование дипломника; |
|  |  | - | ставка взноса в СФР (социальный фонд России), %; |
|  |  | - | ставка взноса на травматизм, %. |

Таким образом, затраты за зарплату руководителя дипломного проекта равны:

Почасовая заработная плата консультанта дипломного проекта равна 379,94 рос. руб. Общее затраченное консультантами время равно 3 часам.

Затраты на зарплату консультантов дипломного проекта () определяются по следующей формуле:

|  |  |  |  |
| --- | --- | --- | --- |
| где |  | - | размер почасовой заработной платы консультанта дипломного проекта; |
|  |  | - | количество часов, которые затрачивает консультант на дипломника. |

Затраты на зарплату дипломных консультантов равны:

Затраты на зарплату дипломника () определяются по следующей формуле:

|  |  |  |  |
| --- | --- | --- | --- |
| где |  | - | размер основной заработной платы, рос. руб; |
|  |  | - | размер дополнительной заработной платы, рос. руб. |

Основная и дополнительная заработная плата дипломника вычисляются по формулам:

,

,

|  |  |  |  |
| --- | --- | --- | --- |
| где |  | - | размер почасовой заработной платы дипломника, рос. руб; |
|  |  | - | процент доплат за труд сверх установленных норм, за изобретательность и так далее. |

Ставка взноса в социальный фонд Российской Федерации и ставка взноса на травматизм для дипломника также составляют 30% и 0.2% соответственно.

Размер почасовой заработной платы дипломника равняется минимальной почасовой заработной плате и равен 109,9 рос. руб. Процент доплат равен 10%.

Затраты на заработную плату дипломника равны:

;

Общие затраты на заработную плату определяются:

Общие затраты на заработную плату равны:

Затраты на электроэнергию () рассчитываются по следующей формуле:

.

|  |  |  |  |
| --- | --- | --- | --- |
| где | Mi | - | паспортная мощность i-го электрооборудования, кВт; |
|  | Ti | - | время работы i-го оборудования за весь период разработки; |
|  | Ц | - | цена электроэнергии, ; |
|  | I | - | вид электрооборудования; |
|  | N | - | количество электрооборудования. |

Паспортная мощность компьютера равна 120 Вт, а монитора - 44 Вт. Цена за кВт час электроэнергии составляет 4,91 рублей.

Расчёт затрат на электроэнергию представлен в таблице 14.

Таблица 14 - Затраты на электроэнергию

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование оборудования | Паспортная мощность, кВт | Время работы оборудования для соответствующей разработки, ч | Цена электроэнергии, | Сумма, рос. руб |
| Компьютер | 0,12 | 110 | 4,91 | 64,81 |
| Монитор | 0,044 | 110 | 4,91 | 23,76 |
| Итоговые затраты на электроэнергию | | | | 88,57 |

К прочим расходам были отнесены затраты на оплату интернета в размере 450 рос. руб., а также на распечатывание документов в размере 300 рос. руб. Таким образом, суммарные затраты на прочие расходы составили 750 российских рублей.

Затраты на общехозяйственные расходы включают заработную плату административно-управленческого персонала организации, охрану организации, отопление и освещение административного здания. Общая сумма общехозяйственных расходов равна 8% от основной заработной платы дипломника, что равно:

Смета затрат на разработку проекта и её структура представлены в таблице 15.

Таблица 15 - Смета затрат на разработку проекта и её структура

|  |  |  |
| --- | --- | --- |
| Статьи затрат | Сумма, рос. руб | Удельный вес, % |
| 1 Амортизация основных фондов | 1 575,62 | 4,35 |
| 2 Затраты на оплату труда | 32 565,92 | 89,82 |
| 3 Затраты на электроэнергию | 88,57 | 0,24 |
| 4 Прочие затраты | 750 | 2,07 |
| 5 Общехозяйственные расходы | 1 274,84 | 3,52 |
| Итого по смете (себестоимость) | 36 254,95 | 100,00 |

На рисунке 8 представлена структура себестоимости в виде круговой диаграммы.



Рисунок 8 - Структура себестоимости

Прибыль рассчитывается по формуле:

Процент рентабельности равен 14. Расчёт прибыли:

Налог на добавленную стоимость рассчитывается по формуле:

.

Согласно статье 164 Налогового Кодекса Российской Федерации ставка НДС составляет 20%. Расчёт налога на добавленную стоимость:

На основе прошлых расчётов составляется цена разрабатываемой программы по следующей формуле:

.

|  |  |  |  |
| --- | --- | --- | --- |
| где | Ц | - | цена разрабатываемой программы; |
|  | С | - | себестоимость; |
|  | П | - | предполагаемая прибыль от реализации; |
|  | НДС | - | налог на добавленную стоимость. |

Расчёт цены разрабатываемой программы:

Для того, чтобы доказать технико-экономическую эффективность разработки был проведён маркетинговый анализ, в ходе которого цена разработки в рамках данного дипломного проекта была сравнена с аналогичными продуктами конкурентов.

Первым конкурентом выступает компания MegaBotCompany, которая предлагает разработку базы данных за 60 тысяч рос. руб. В цену не входит разработка пользовательского и прикладного программного интерфейса для базы данных.

Вторым конкурентом является компания Micro-Solution, готовая создать базу данных MySQL за 53 тысяч рос. руб. В цену также не входит разработка пользовательского и прикладного программного интерфейса.

Третьим конкурентом является компания GeneralITCompany, которая готова создать базу данных, пользовательский и прикладной программный интерфейс за 110 тысяч рос. руб.

Сравнение цены разработки в рамках дипломного проекта и цены конкурентов приведены в таблице 16.

Таблица 16 - Маркетинговый анализ

|  |  |  |  |
| --- | --- | --- | --- |
| Конкурент | Цена конкурента, рос. руб. | Цена разработчика, рос. руб. | Отклонение цены разработчика от цены конкурента, рос. руб. |
| MegaBotCompany | 60 000 | 49 596,77 | -10 403,23 |
| Micro-Solution | 53 000 | 49 596,77 | -3 403,23 |
| GeneralITCompany | 110 000 | 49 596,77 | -60 403,23 |

Таким образом, данный программный продукт является конкурентноспособным и его разработка является эффективной с технико-экономической точки зрения.

# ГЛАВА 4 МЕРОПРИЯТИЯ ПО ТЕХНИКЕ БЕЗОПАСНОСТИ

Охрана труда - это комплексная система мероприятий, которая включает в себя различные аспекты, направленные на обеспечение безопасности и здоровья работников в процессе их трудовой деятельности. Это особенно важно для профессий, которые требуют длительного времени в сидячем положении, таких как программисты или любые другие профессии, связанные с работой за компьютером.

Одним из ключевых элементов охраны труда является правильная организация рабочего места. Компьютерные станции должны быть настроены таким образом, чтобы минимизировать риск возникновения мускулоскелетных заболеваний и усталости глаз. Это включает в себя правильную высоту стола и стула, а также оптимальное расположение монитора и клавиатуры. Рабочее место должно быть удобным и функциональным, чтобы работник мог эффективно выполнять свои обязанности без риска для здоровья.

Крайне важно проводить мероприятия по инструктированию работников о правильных методах работы с компьютером. Это включает в себя обучение правильной позе тела, частоте перерывов для отдыха и упражнений для глаз. Это помогает предотвратить возможные травмы и заболевания, связанные с монотонной работой за компьютером. Регулярные тренинги и семинары по охране труда могут помочь работникам освоить правильные методы работы и уменьшить риск возникновения профессиональных заболеваний.

Противопожарная безопасность - это ещё один важный аспект охраны труда. Она включает в себя ряд мероприятий, направленных на предотвращение возгорания и обеспечение безопасной эвакуации в случае пожара. Это включает в себя знание расположения пожарных выходов, путей эвакуации и местоположения пожарных средств, таких как огнетушители. Работники должны быть обучены действиям в случае пожара и знать, как использовать пожарное оборудование.

Другим важным фактором для поддержания пожарной безопасности является регулярная проверка проводных соединений и оборудования, которые обладают потенциалом становления причиной возникновения пожара. Также важно регулярно проводить проверки систем пожарной безопасности, чтобы в случае пожара минимизировать нанесённый урон. Это включает в себя проверку состояния электрических проводов, наличие и исправность пожарных датчиков и сигнализаций.

Частью мероприятий по технике безопасности являются и меры по охране окружающей среды. Эти меры включают в себя действия по минимизации негативного воздействия на окружающую среду в процессе обучения и работы. Это может включать в себя использование энергоэффективного оборудования и технологий, чтобы сократить потребление электроэнергии. Это также может включать в себя использование зелёных технологий и практик, таких как использование солнечной энергии и утилизация отходов.

Важно также регулярно проводить мероприятия по утилизации и переработке отходов, а также содействовать использованию электронных документов вместо бумажных, чтобы уменьшить потребление бумаги и лесных ресурсов. Это может включать в себя разработку и внедрение систем управления отходами, которые помогают минимизировать объем отходов и улучшить их утилизацию.

Общие принципы охраны труда, противопожарной безопасности и охраны окружающей среды являются основой безопасной и устойчивой рабочей среды для работников. Это помогает создать условия, при которых работники могут выполнять свои обязанности эффективно и безопасно. Создание безопасной и здоровой рабочей среды - это не только обязанность работодателя, но и важная часть корпоративной культуры, которая способствует улучшению производительности и настроению работников.

# ЗАКЛЮЧЕНИЕ

В рамках данного проекта была разработана база данных, представляющая собой основу для эффективного функционирования веб-ресурса, направленного на обучение английскому языку студентов компьютерных специальностей. Созданный пользовательский интерфейс в виде веб-страницы обеспечивает комфортное взаимодействие с базой данных, что позволяет легко находить и обрабатывать необходимый материал. Прикладной программный интерфейс стандартизирует обмен данными между компонентами системы, что обеспечивает их эффективную интеграцию и совместную работу.

База данных нацелена на хранение данных по модульному принципу , что обеспечивает гибкость в организации учебного процесса и возможность повторного использования ранее созданных модулей и тем.

Данные модули являются важным шагом в разработке веб-ресурса, способствующего качественному обучению английскому языку студентов компьютерных специальностей, улучшению их навыков и повышению их конкурентоспособности в современном мире образования.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Evans V., Doolye J., Pontelli E. Software Engineering – Numbury: Express Publishing, 2014. – 116 с.
2. Evans V., Dooley J., Wright S. Information Technology – Newbury: Express Publishing, 2011. – 122 с.
3. Кузнецов М. Самоучитель MySQL 5 – БХВ - Петербург, 2014 г. – 536с.: ил.
4. Маклафлин Б. – PHP и MySQL. Исчерпывающее руководство. – СПб.: Питер, 2013. – 512 с.: ил.
5. Уроки PHP. По материалам форума forum.php.su [Электронный ресурс] – http://www.php.su/lessons/.